Quadrilateral Mesh Smoothing

Xuan Huang

Advised by Professor Dianna Xu

March 7, 2018

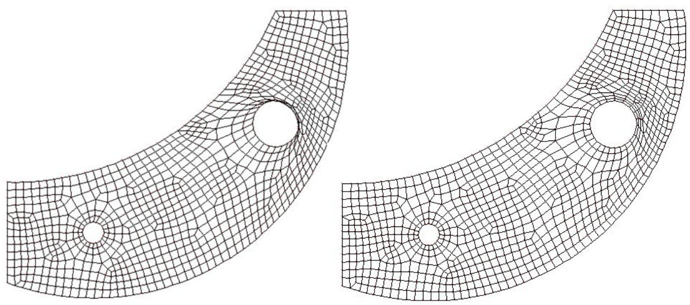


Figure 1: Quadrilateral meshes smoothing via vertex repositioning

1 Introduction

Mesh quality improvement is an important problem with a wide range of practical applications. Element quality of a mesh heavily affects the results of numerical simulation, such as rendering (Figure 2) or physical simulation (Figure 3) using that mesh. In the context of finite element mesh smoothing, vertex repositioning is the primary technique employed, where we allow vertex motion only and the connectivity of all edges remains unchanged. In the 2D cases we are interested in, this means that only the x y values of vertices can be modified. Figure 2 and Figure 3 are very helpful for introducing these concepts!

Based on the ideas of Laplacian Smoothing and our previous work on triangular mesh smoothing, this paper presents an easy-to-implement, computationally inexpensive method to provide quality improvements on any given quadrilateral mesh. better to say "in this paper, we present"

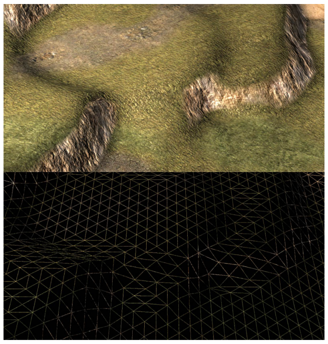


Figure 2: A example of terrain rendering

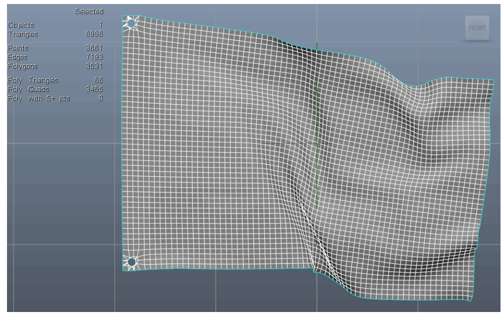


Figure 3: High-quality cloth simulation with NVIDIA

2 Element Quality

Element quality is measured either by max/min angles or aspect ratio (longest edge over shortest), or both. In a quadrilateral mesh the aspect ratio is defined as the largest ratio of longest edge over shortest edge among all elements, and max/min angles are the maximum/minimum among all angles. These two proprieties are related in triangles (by the law of sine), but the relationship is quite loose in quads (Figure 4). We investigate a smoothing method based on the idea of improving aspect ratio by circumscircles. good definition

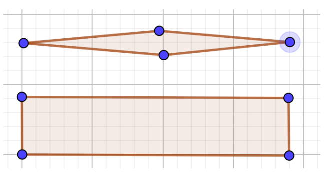


Figure 4: Aspect ratio and angles are not strictly related

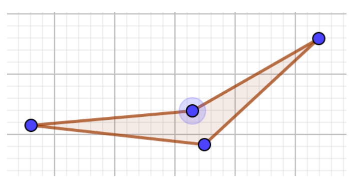


Figure 5: Non-convex quad

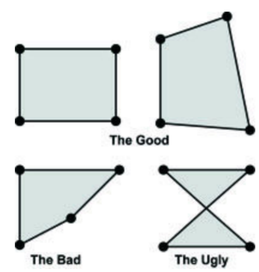


Figure 6: The good, the bad, and the ugly

Convexity is another important aspect of quadrilateral elements (Figure 5). It is very undesired to generate any element that has an angel greater than 180 degree. Again this is never a problem for triangles, and one of our goals will be trying to eliminate such cases from happening. In addition, self-intersection and edge flipping should be avoided under any circumstances. Figure 6 illustrates a general situation for desired and undesired quadrilateral elements. concise and clear

3 Background

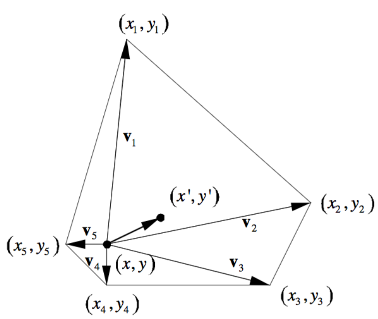
Maybe explain this in a little bit more detail: 1-2 more sentences

Existing methods usually divide into mesh modification via vertex insertion/deletion, edge/face swapping and remeshing [Dey and Ray 2010], or vertex repositioning without changing mesh connectivity [Amenta et al. 1997; Field 1988; Zhou and Shimada 2000]. Among those that do not modify mesh topology, Laplacian smoothing is the most commonly used because of its simplicity. In its most basic form, it moves each vertex to the centroid of the polygonal region formed by its neighboring vertices. It is a local method with very low computational cost, compared to the alternatives, which are optimization-based [Chen and Holst 2011; Freitag 1997; Parthasarathy and Kodiyalam 1991]. The most closely related work is Zhou and Shimadas angle-based Laplacian smoothing [Zhou and Shimada 2000], which is a variant of Laplacian smoothing. add in the annotation(which is discussed in detail in Section 4.1)

4 Triangular Mesh Smoothing

4.1 Laplacian and its Angle Based Variation

The widely used Laplacian smoothing method is a computationally inexpensive, easy to implement local technique for triangular mesh smoothing. It simply moves each vertex to the centroid specified by the polygon formed by all it adjacent vertices, which is also known as the star region [Amenta et al. 1997; Field 1988; Zhou and Shimada 2000]. The process is applied on every vertices of the mesh for several iterations, usually until the positions of all vertices converges or the mesh quality no longer improves.



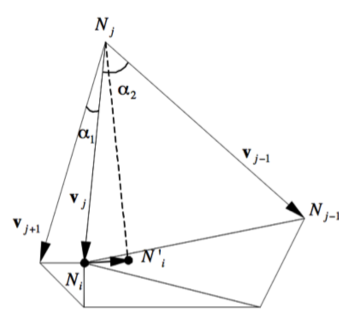


Figure 7: **Left**: The Laplacian method; **Right**: The angle-based variation. Rotating edge N_jN_i to angle bisectors to get the new position N_i' for vectex N_i

Problem arises for this method as it does not always move the vertices to optimal positions, and due to the fact that the centroid of a polygon could lie very close to, or even outside the shape itself, sometimes it generate invalid elements.

Zhou and Shimada presents an angle-based smoothing method based on the Laplacian smoothing algorithm [Amenta et al. 1997; Field 1988; Zhou and Shimada 2000]. Instead of taking the centroid to be the new position of each vertex, they rotate each interior edge attached to the angle bisector, and take the average value resulted from all such repositioning actions to be the new position of the vertex.

Figure 8 shows the Laplacian method and its variation with a specific example. In the star region of the vertex to be smoothed, the Laplacian method calculated the centroid of the star region and assign the vertex position there. The angle-based variation rotated the interior edges to find the new position, and take the average in the end.

The result has a general improvements in mesh quality improving comparing to the original Laplacian method, and has largely reduced the possibility of getting invalid elements. It also provides a mechanism to implement the method on quad meshes simply by treating the quadrilateral element as the combination of two triangles.

4.2 Aspect-Ratio Based

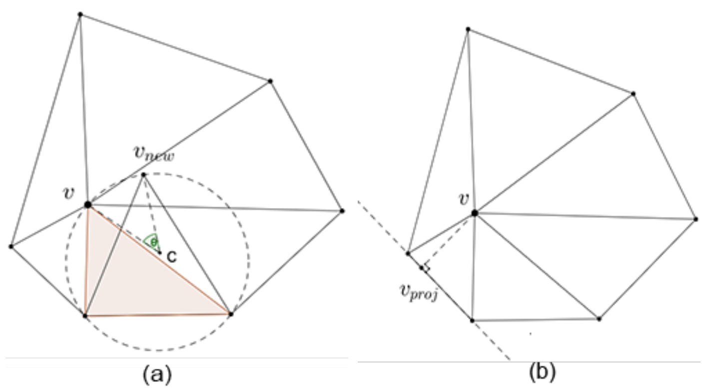


Figure 8: (a) Move v to v_{new}; (b) $\overline{vv_{proj}}$ is the shortest distance to star boundary

This section introduce our previous method to improve aspect ratio as an alternative to the existing angle-based method. We believe that aspect ratio will lead to a more balanced improvement in triangles, and this is the work motivated by aspect ratio improvements for quadrilateral meshes, which are unrelated to angles. Clear justification

The idea is that inside the star region of each vertex, move the center vertex along the circumcircle of each incident triangle in a direction that will decrease the difference between each pair of adjacent interior edges. Each move must improve the aspect ratio of a triangle, because one of the two edges must be the shortest or longest edge of the triangle considered. Some restrictions are applied so that triangles already with good quality are not further modified and potentially sacrificed for worse ones.

The algorithm performs the following steps:

1. As shown in Figure 8, for each vertex v not on the boundary there are k faces inside its star polygon. For each triangle if the two interior edges has a ratio higher than 1.5, calculate its circumcircle centered on C.

2. Move v clockwise or counter-clockwise towards the longer incident edge.

3. Let $\overline{vv_{proj}}$ be v's vertical distance to each star polygon edge. If the new position ends up decreasing the shortest distance from the vertex to the star polygon boundary, discard movement.

4. Take the mean of all positions after processing all k faces and assign the v there. Iterate until the resulting positions converges.

The max aspect ratio is improved over angle-based method in all cases tested. The algorithm might also provide some additional improvements on angles. It tends to recover seriously distorted areas and is also much less likely to result in invalid shapes compared to the angled- based method.

4.3 Results

The algorithm typically generate a mesh with aspect ratio under 3 as well as some improvements on angle. It tends to recover seriously distorted areas and very unlikely to result in invalid shape. Followed are 4 examples, column (a) represents the original mesh, (b) represents the result from Angle-based smoothing, and (c) represents our result from Aspect-ration based smoothing, followed by a static comparison table :

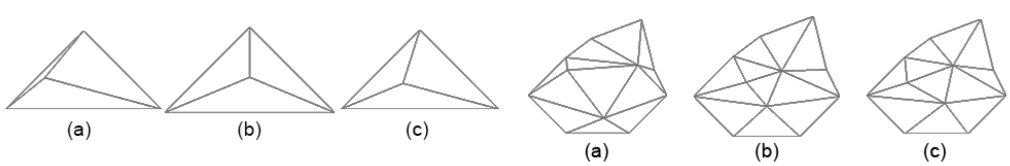


Figure 9: simple triangular meshes: 3 triangles and 10 triangles

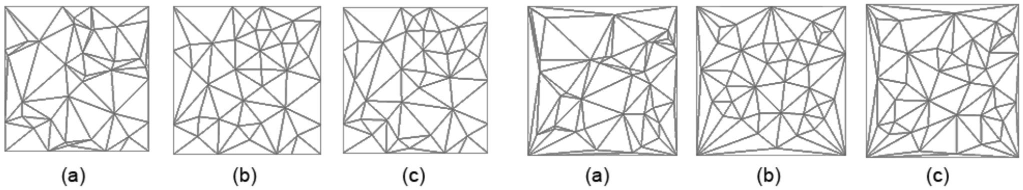


Figure 10: 40 triangles. A nice input example and a bad one.

	Original	Angle-based	Aspect Ratio-based
Simple Triangle	max: 168.465 min: 5.19443 max AR: 3.12348	**max: 135** **min: 22.5** max AR: 2.41421	max: 142.597 min: 14.9712 **max AR: 2.35129**
12 faces	max: 133.668 min: 10.0946 max AR: 5.70305	**max: 115.484** min: 22.6975 max AR: 2.33943	max: 133.796 **min: 22.969** **max AR: 2.14802**
40 faces (1)	max: 162.387 min: 2.48955 max AR: 11.1803	**max: 132.203** **min: 18.5365** max AR: 2.82876	max: 159.777 min: 8.16759 **max AR: 2.56345**
40 faces (2)	max: 174.549 min: 1.63658 max AR: 13.0384	max: 179.502 min: 0.248059 max AR: 3.42261	**max: 174.147** **min: 2.38678** **max AR: 2.66102**
1000 faces	max: 168.663 min: 1.03697 max AR: 48.8833	**max: 138.854** min: 1.23549 max AR: 35.5843	max: 160.885 **min: 2.20761** **max AR: 19.4623**

Figure 11: comparison table

As shown the aspect ratio tend to beat angle-based method at most cases with also a slight improvement on angles. Under extreme cases the aspect ratio method also protects small angles. !!! I think it would be nice to summarize the result and give more detailed discussions on these data

5 Quadrilateral Mesh Smoothing

5.1 Previous work and its limitation on Quadrilateral meshes

Give an example what do you mean by loose angle-aspect ratio relationship
Either explain it briefly but I guess it's easier to show one example

The triangular mesh smoothing methods do provide techniques via cutting the quads into two triangles and smooth the resulted triangular mesh. Apparently this way of smoothing ignores the proprieties special to quads, such as the loose angle-aspect ratio relationship, and the fact that a quad could become non-convex during the process. We would like our algorithm to be aware of such proprieties, and provide a more reliable solution to quadrilateral meshes specifically.

This sentense doesn't make much sense to me. I might need more background knowledge. What's a parametric spaces

Try reorganize and rewording this paragraph

Other methods including constraining vertices with parametric spaces derived from individual mesh elements (faces, edges) and optimizing an objective function with respect to the parametric coordinates [Rao and Mikhail], cleaning-up nodes with higher number of connectives. In the meantime, there is no universal consensus on the measurement of quadrilateral elements. A list of metrics encoding information of the element size, shape, and skew are used to analyze the quality of quadrilateral elements [Patrick M. Knupp].

Also breaking up the sentence and explain with examples might help.

How do you determine the skew?

Our algorithm design focuses on the worse angles and aspect-ratio in the mesh, since they are usually the bottleneck in terms of further simulation work. We propose a method to improve both angle and aspect-ration locally in each quad, and then avoid the problem of self-intersecting and non-convexity by a distance check mechanism within each star region. The procedure is repeated until there is no further improvements.

5.2 Our method

After sorting all incoming quads by their maximum angles, we smooth the mesh starting from the worst one. The algorithm performs the following steps: Organizing into steps really help with the explanation

how do you determine which is the worst one?

1. As shown in Figure 12, for each vertex v not on the boundary we look at its angle within the quad, push the v further in the direction perpendicular to the diagonal if the angle is larger than a certain $maxang$, and pull v closer in the direction perpendicular to the diagonal if the angle is smaller than a certain $minang$. Define it

2. For the two adjacent vertices of v not on the boundary, shrink (if large angle) or elongate (if small angle) the diagonal accordingly.

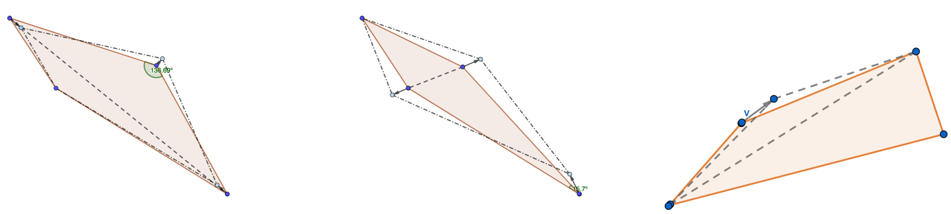


Figure 12: Smoothing scheme: Large angle, Small angle, and AR improvement

3. Move v parallel to the diagonal towards the longer incident edge.

4. Let $\overline{vv_{proj}}$ be v's vertical distance to each star polygon edge as well as to each quad's diagonal that doesn't include v. If the new position ends up decreasing the shortest distance, discard movement.

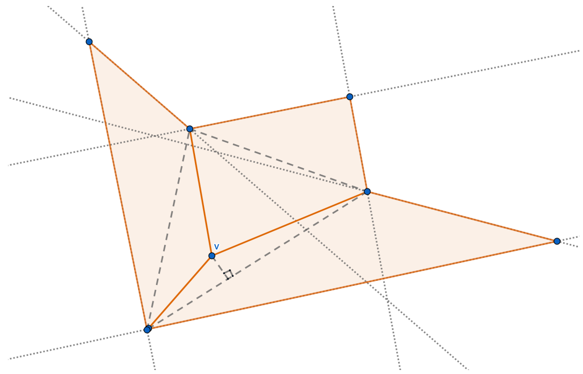


Figure 13: Boundary distance check in step 4

5. Take the mean of all positions after processing all quads (not yet, still implementing)
6. Iterate until the resulting positions converges. define convergence criterion?

Step 1-3 provide intuitive way to improve the two factors of individual quad element. The shortest distance check in step 4 serves as a protection against non-convexity and self-intersection, since the vertex is not allowed to get too close to any diagonal or quad boundary.

5.3 Result and Further Plan

We apply the method on single quads as well as some complicated quadrilateral meshes. There is a general improvements on both the max/min angle and the aspect ratio, while no non-convex or self-intersecting elements are found up to this point.

There seems to be a tendency of sacrificing small angles to improve the large ones (see the example chart Figure 15 below), which is reasonable since it is often the large angles that per definition-wise, do you mean bad? cause the extremely ugly cases, and our algorithm specifically restricts such circumstances from appearing. But still there may be some room for tuning, as there are several hard-coded coefficients.

The next step, while investigating the problem above, is to take the average as indicated in step 5, which we think will probably provide a more balanced result. The other issue is that whether it is the case that when the distance check stops the smoothing movements, there is truly no possible improvements. Then we may need a slightly more global way of identifying the quality of the quads and their surrounding regions. Other measurements (likely the CRE method) would be applied if time allows.

What does CRE stands for and I didn't remember seeing it explained in background

Comparison images (example here)

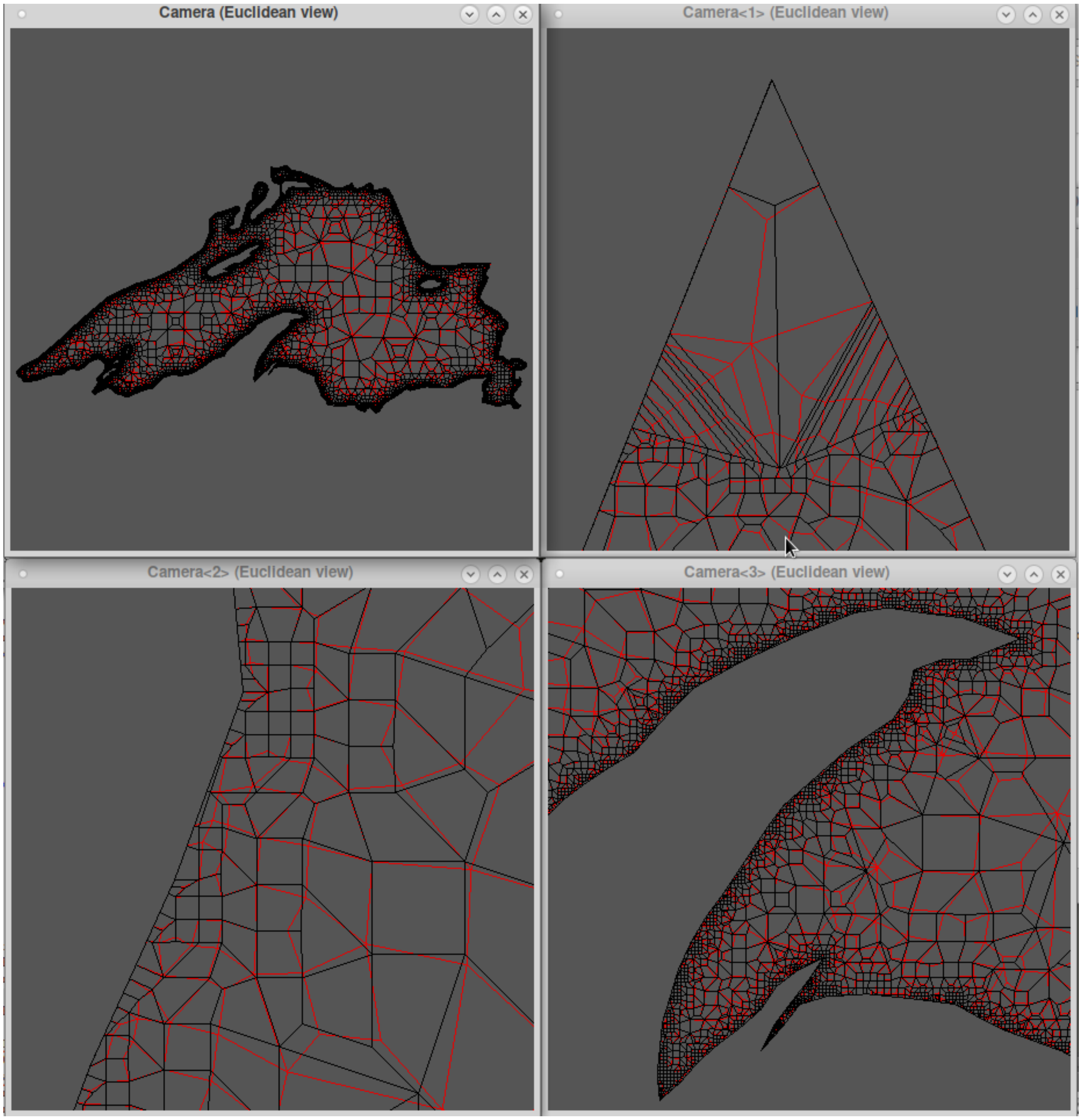
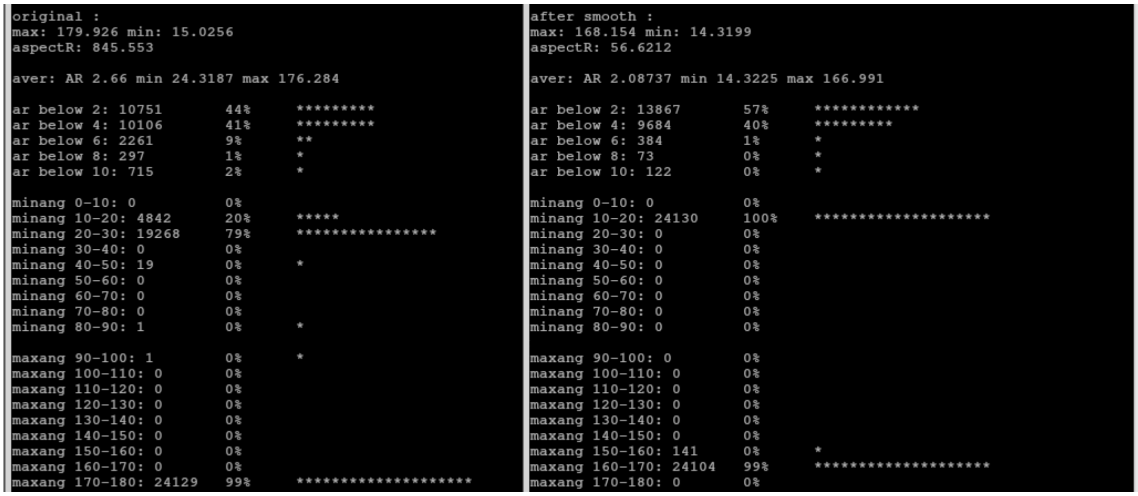


Figure 14: 24K quads example - lake superior (Black-input Red-result)

Again, you want to discuss your result (both the figure and comparison tables and charts for the readers,
as we might not be familiar what each line color means and what each section of the comparison chart refers to?

It would also be nice to reiterate the ideas(your algorithm, some concepts etc) using this example

Comparison tables and charts (example here)



```
original :
max: 179.926 min: 15.0256
aspectR: 845.553

aver: AR 2.66 min 24.3187 max 176.284

ar below 2: 10751       44%     *********
ar below 4: 10106       41%     *********
ar below 6: 2261        9%      **
ar below 8: 297         1%      *
ar below 10: 715        2%      *

minang 0-10: 0          0%
minang 10-20: 4842      20%     *****
minang 20-30: 19268     79%     ****************
minang 30-40: 0         0%
minang 40-50: 19        0%      *
minang 50-60: 0         0%
minang 60-70: 0         0%
minang 70-80: 0         0%
minang 80-90: 1         0%      *

maxang 90-100: 1        0%      *
maxang 100-110: 0       0%
maxang 110-120: 0       0%
maxang 120-130: 0       0%
maxang 130-140: 0       0%
maxang 140-150: 0       0%
maxang 150-160: 0       0%
maxang 160-170: 0       0%
maxang 170-180: 24129   99%     ********************
```

```
after smooth :
max: 168.154 min: 14.3199
aspectR: 56.6212

aver: AR 2.08737 min 14.3225 max 166.991

ar below 2: 13867       57%     ************
ar below 4: 9684        40%     *********
ar below 6: 384         1%      *
ar below 8: 73          0%      *
ar below 10: 122        0%      *

minang 0-10: 0          0%
minang 10-20: 24130     100%    ********************
minang 20-30: 0         0%
minang 30-40: 0         0%
minang 40-50: 0         0%
minang 50-60: 0         0%
minang 60-70: 0         0%
minang 70-80: 0         0%
minang 80-90: 0         0%

maxang 90-100: 0        0%
maxang 100-110: 0       0%
maxang 110-120: 0       0%
maxang 120-130: 0       0%
maxang 130-140: 0       0%
maxang 140-150: 0       0%
maxang 150-160: 141     0%      *
maxang 160-170: 24104   99%     ********************
maxang 170-180: 0       0%
```

Figure 15: 24K quads example - lake superior (runs within a minute)

Other analysis method applied (to do)

6 Documentation

L. Chen and M. Holst. 2011. Efficient Mesh Optimization Schemes based on Optimal Delaunay Triangulations. Computer Methods in Applied Mechanics and Engineering 200 (2011), 967984.

T. Dey and T. Ray. 2010. Polygonal Surface Remeshing with Delaunay Refinement. Engineering with Computers 26 (2010), 298301.

D. Field. 1988. Laplacian Smoothing and Delaunay Triangulations. Communications in Applied Numerical Methods 4 (1988), 709712. Issue 6.

L. Freitag. 1997. On Combining Laplacian and Optimization-based Mesh Smoothing Techniques. AMD Trends in Unstructured Mesh Generation 220 (1997), 3743.

V. Parthasarathy and S. Kodiyalam. 1991. A Constrained Optimization Approach to Finite Elemement Mesh Smoothing. Finite Elements in Analysis and Design 9 (1991), 309320.

T. Zhou and K. Shimada. 2000. An Angle-based Approach to Two-dimensional Mesh Smoothing. In Proceedings, 9th International Meshing Roundtable. 373384.

J. Robinson. 1987 "CRE method of element testing and the Jacobian shape parameters", Engineering Computations, Vol. 4 Issue: 2, pp.113-118, https://doi.org/10.1108/eb023689

P M. Knupp. Algebraic mesh quality metrics for unstructured initial meshes. Parallel Computing Sciences Department, Sandia National Laboratories. 2001.

Figure 1. An example of quadrilateral mesh smoothing. Reprinted from Mesh Smoothing Challenges in the Industry, In Slide Player, N. Mukherjee, Retrieved February 17, 2018, from http://slideplayer.com/slide/5853324/. Copyright 2016 by N. Mukherjee.
Figure 2. An example of terrain rendering. Reprinted from Terrain texture projection artifacts, in Developer's blog for IceFall Games, Retrieved February 17, 2018, from http://cgicoffee.com/blog/2016/08/nvidia-flex-cloth-simulation, mtnphil. Copyright September 25, 2012 by mtnphil at Developer's blog for IceFall Games.
Figure 3. High-quality cloth simulation with NVIDIA. Reprinted from Maxwell Render/Learn/Render/Channel Rendering, in Next Limit, Retrieved February 17, 2018, from http://support.nextlimit.com/display/rf2016docs/Channel+Rendering. Copyright by RealFlow 10 Documentation.
Figure 4-5. Self made with GeoGebra at https://www.geogebra.org/.
Figure 6. The good the bad and the ugly. Reprinted from Plate/Shell in risa.com, Retrieved February 17, 2018, from https://risa.com/risahelp/risa3d/Content/3D_2D_Only_Topics/Plates.htm. Copyright by risa.com.
Figure 7 from T. Zhou and K. Shimada [2000].
Figure 8-15. Self made with GeoGebra at https://www.geogebra.org/ and Meshlab, modified with Adobe Photoshop.